Microsoft

.net developer training

Create XML Web Services and Applications with Visual Studio .NET

Application Interoperability

Module 14 Agenda

- Introduction
- .NET Using COM
- Wrapper Assemblies
- COM Using .NET
- Platform Invoke
- Performance Considerations

Introduction

- Preserve your investment in your existing code
  - No need to start over
  - Use existing COM components and COM+ services
- Incremental migration path
  - Migrate your application step-by-step
- The reality is that some things never change
  - Necessary to interoperate with code that is unable to change

Introduction
Old Versus .NET

Unmanaged Code

- Binary standard
- Type libraries
- Immutable
- Type unsafe
- Interface-based
- HResults
- Guids

Managed Code

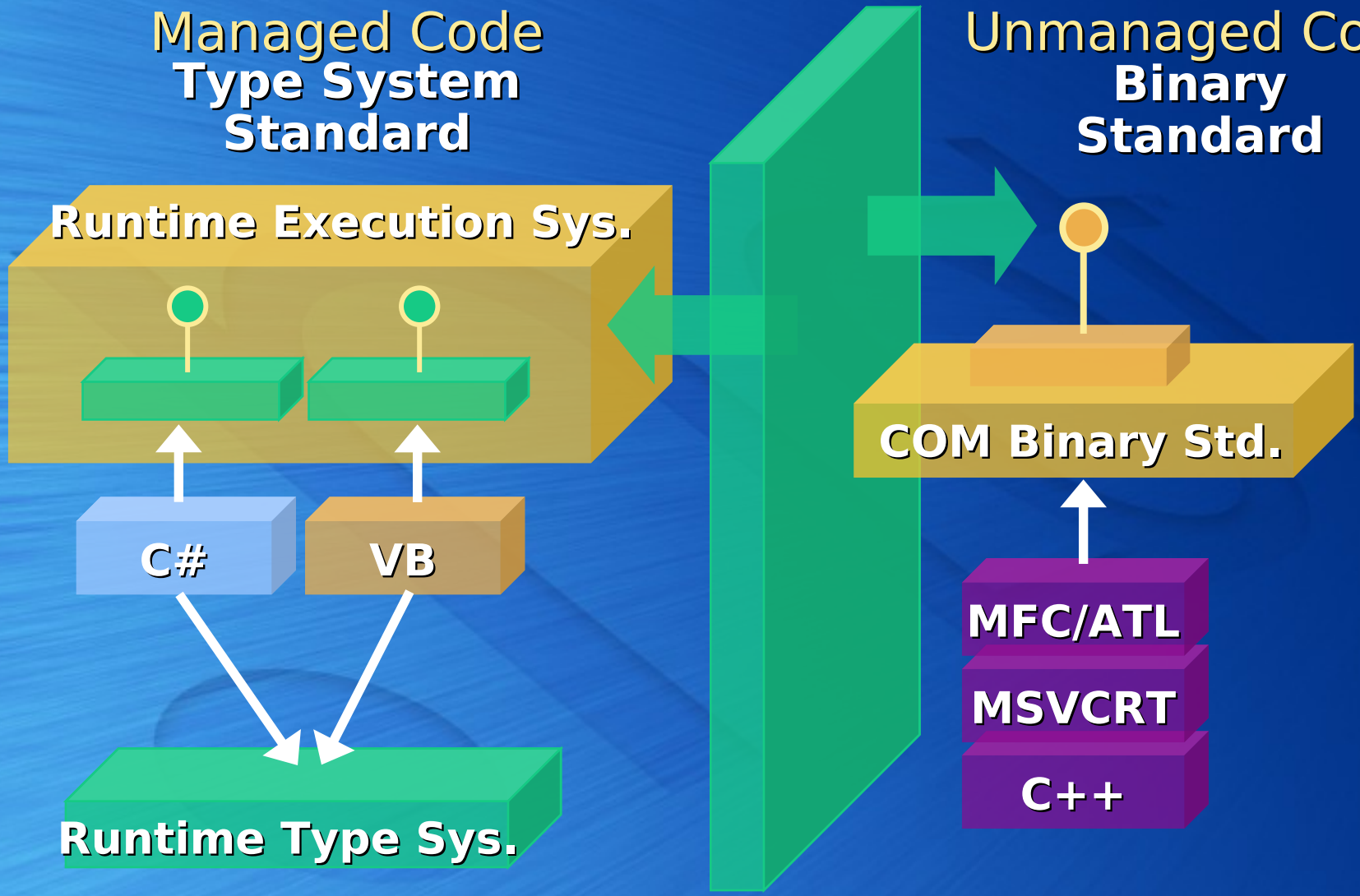
- Type standard
- Assemblies
- Resilient bind
- Type safe
- Object-based
- Exceptions
- Strong names

Introduction
Bi-directional COM Interop

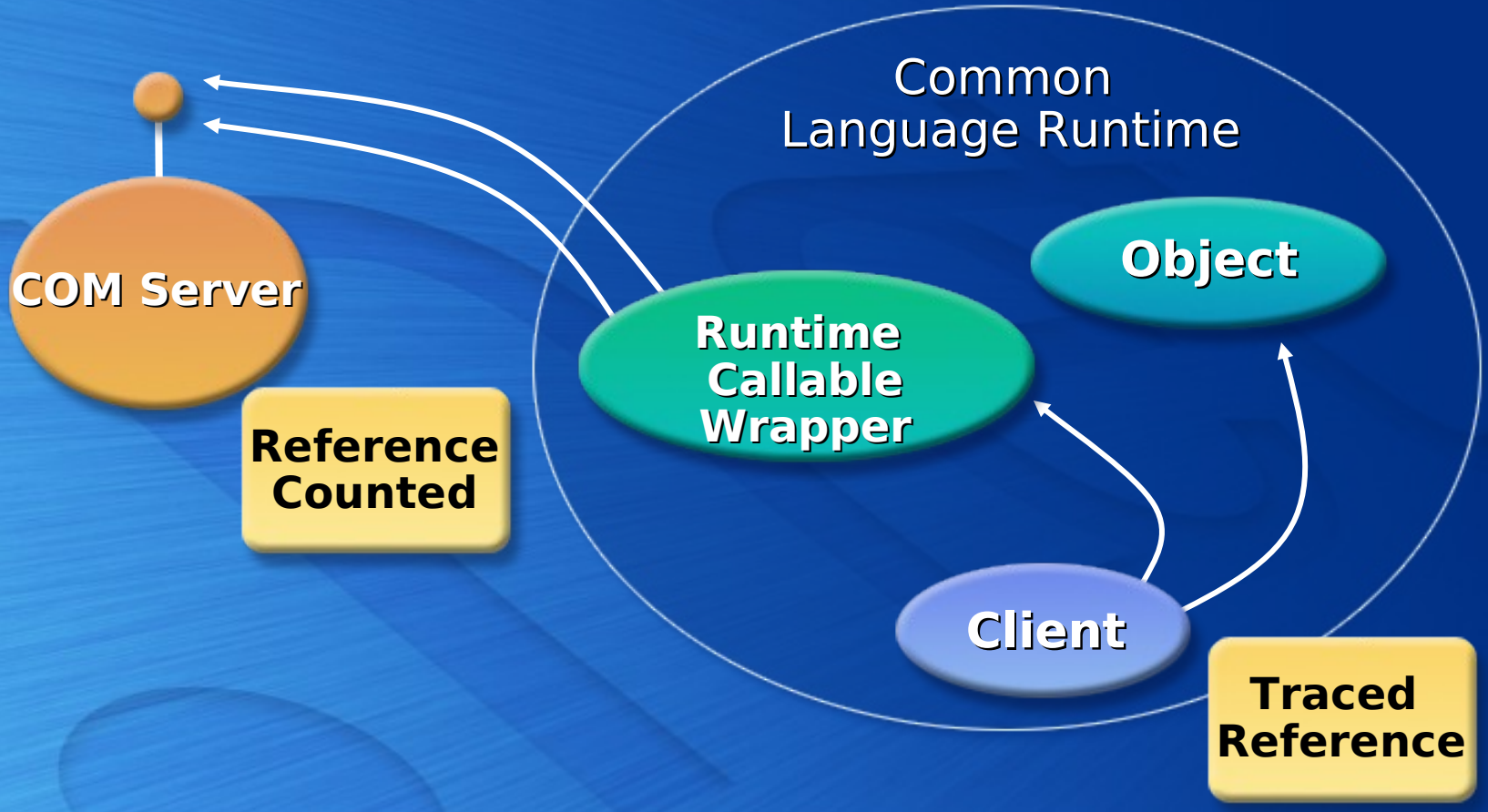
- .NET Using COM
  - Allows .NET types to access COM types

- COM Using .NET
  - Allows COM types to access .NET types

.NET Using COM
Integration Services



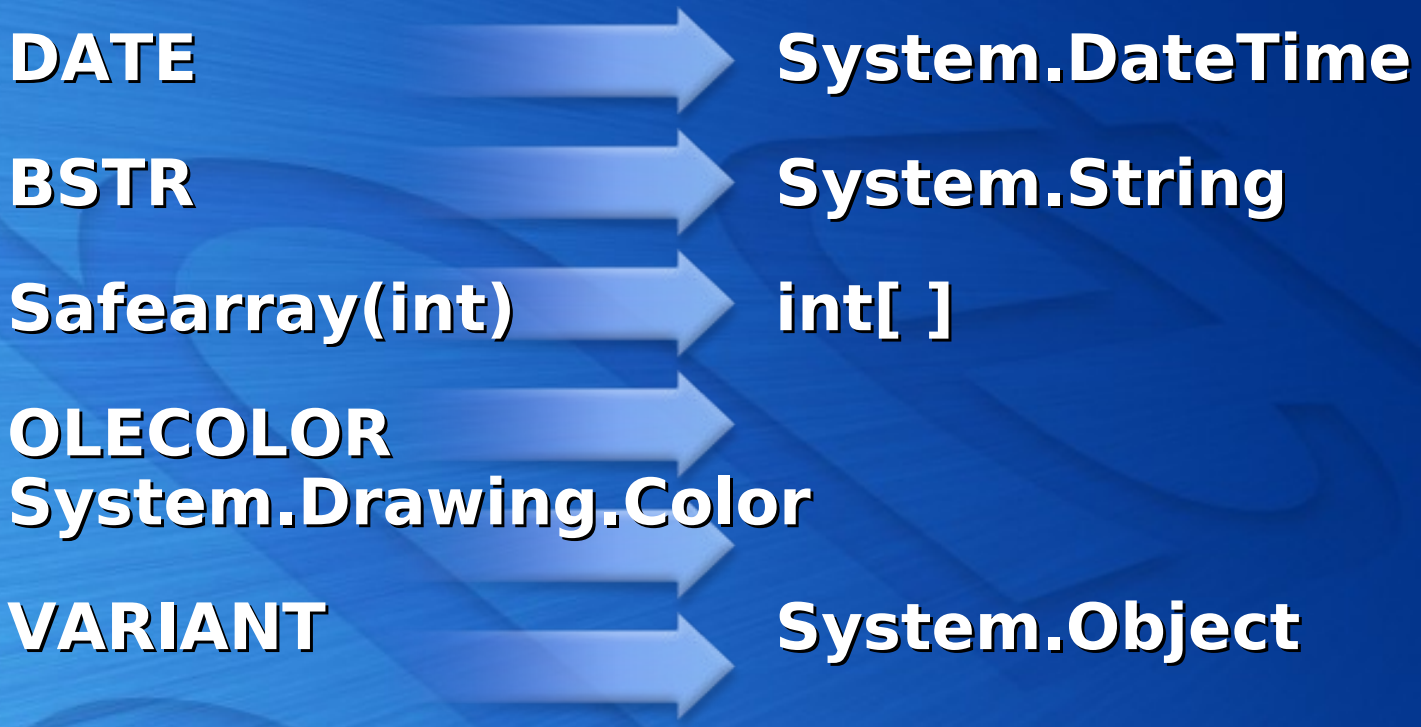
.NET Using COM Interop



.NET Using COM
Using COM Types from .NET

- Create an assembly containing type definitions for COM types
  - By using type library importer (TlbImp)
    - TlbImp MyLib.tlb /out:MyLib.Dll
  - By adding references in Visual Studio®
- Use the types as managed types
  - Create instances with new
  - Catch exceptions
  - Extend exceptions

Data Type Conversion
Performed by TlbImp



Using COM Types from .NET

- Failed HResults are automatically mapped to exceptions
- Enumerations are automatically mapped to IEnumVariant
- ConnectionPoints are automatically mapped to delegates

Installing Wrapper Assemblies

- Assembly with COM type definitions must be resolvable at runtime
  - Installed in the application directory
    - Copy assembly along with the application file
  - Installed in the Global Assembly Cache
    - Generate key file in Visual Studio® .NET
    - Sign the assembly using TlbImp
      - C:\>TlbImp MyLib.tlb /keyfile:MyKey.snk

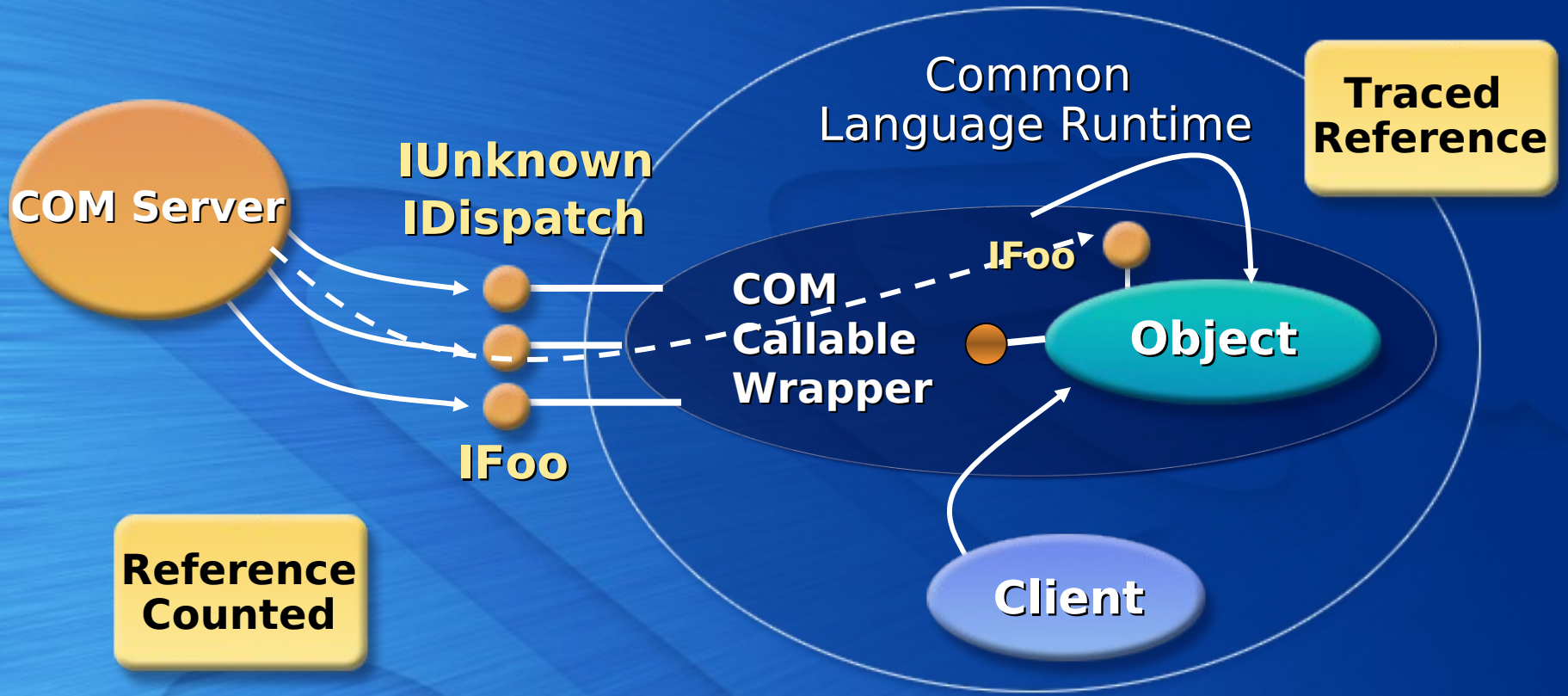
COM Components

- Components called via COM must be registered
- Runtime creates a COM Callable Wrapper (CCW)
  - Reference counted like COM Object
  - Transparent to assembly
  - Handles marshalling

Using COM Classes Inside .NET
Demo

- COM Class From .NET Demo

COM Using .NET
Interoperability



COM Using .NET
Using .NET Types from COM

- Create, manage type in the usual way
  - Only the public are exposed to COM
  - Classes must have default ctor to be creatable
- Use TlbExp to create COM type library
- Register the assembly with RegAsm
  - Only needed for Interop
- Reference the typelib as necessary
- Use the types as normal COM types

Platform Invoke (P/Invoke)

- Provides access to static entry points in unmanaged DLLs (Win32® API)
- Similar to:
  - Visual Basic® Declare statement
  - LoadLibrary/GetProcAddress
- C# requires a method definition with a custom attribute (VB uses Declare)
- Uses the same underlying marshalling service as COM Interop

Performance Considerations

- Transitions have overhead
  - Roughly 20-30 instruction per call
- Data marshalling adds overhead
  - Depending on type and size of data
  - Blittable types are inexpensive
- Make transitions wisely
  - Chunky calls as opposed to chatty

Summary

- Existing components can be used in .NET
- Applications can mix .NET and existing features
- Win32 and other APIs can be used from .NET

Resources

- ***Interoperating with Unmanaged Code*** **- Simple overview of interoperability. There are two links on this page as well** **http://msdn.microsoft.com/library/dotnet/cpguide/cpconinteroperatingwithunmanagedcode.htm**
- ***Handling Language Interoperability with the Microsoft .NET Framework*** **- This article looks at the Common Language Runtime and the Common Language Specification as well as metadata**
- **http://msdn.microsoft.com/library/techart/InteropdotNET.htm**

In-Depth Training on Interoperability

Direct from Microsoft

- 2349A: Programming the Microsoft .NET Framework with C# (Prerelease)
- 2124A: Introduction to C# Programming for the Microsoft .NET Platform (Prerelease)
  - Available now on Visual Studio .NET Beta 1

Microsoft®

14.22